PRIOR-GUIDED BAYESIAN OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

While Bayesian Optimization (BO) is a very popular method for optimizing expensive black-box functions, it fails to leverage the experience of domain experts. This causes BO to waste function evaluations on bad design choices (e.g., machine learning hyperparameters) that the expert already knows to work poorly. To address this issue, we introduce Prior-guided Bayesian Optimization (PrBO). PrBO allows users to inject their knowledge into the optimization process in the form of priors about which parts of the input space will yield the best performance, rather than BO's standard priors over functions (which are much less intuitive for users). PrBO then combines these priors with BO's standard probabilistic model to form a pseudo-posterior used to select which points to evaluate next. We show that PrBO is around 12x faster than state-of-the-art methods without user priors and $10{,}000\times$ faster than random search on a common suite of benchmarks, and achieves a new state-of-the-art performance on a real-world hardware design application. We also show that PrBO converges faster even if the user priors are not entirely accurate and that it robustly recovers from misleading priors.

1 INTRODUCTION

Bayesian Optimization (BO) is a data-efficient method for the joint optimization of design choices that gained great popularity in recent years. It is impacting a wide range of areas, including hyperparameter optimization (Snoek et al., 2012; Falkner et al., 2018), AutoML (Feurer et al., 2015a; Hutter et al., 2018), robotics (Calandra et al., 2016), computer vision (Nardi et al., 2017; Bodin et al., 2016), environmental monitoring (Marchant & Ramos, 2012), combinatorial optimization (Hutter et al., 2011), experimental design (Azimi et al., 2012), RL (Brochu et al., 2010), Computer Go (Chen et al., 2018), hardware design (Koeplinger et al., 2018; Nardi et al., 2019) and many others. It promises greater automation so as to increase both product quality and human productivity. As a result, BO is also established in many large tech companies, e.g., with Google Vizier (Golovin et al., 2017) and Facebook BoTorch (Balandat et al., 2019).

Nevertheless domain experts often have substantial prior knowledge that standard BO cannot incorporate. Users can incorporate prior knowledge by narrowing the search space; however, this type of hard prior can lead to poor performance by missing important regions. BO also supports a prior over functions $p(f)$, e.g., via a kernel function. However, this is not the prior experts have: users often know which ranges of hyperparameters tend to work best, and are able to specify a probability distribution $p_{\text{best}}(\boldsymbol{x})$ to quantify these priors. E.g., many users of the Adam optimizer (Kingma & Ba, 2015) know that its best learning rate is often in the vicinity of 1e-3 (give or take one order of magnitude), yet one may not know the accuracy one may achieve in a new application. Similarly, Navruzyan et al. (2019) derived neural network hyperparameter priors for image datasets based on their experience with five datasets. In these cases, users know potentially good values for a new application, but cannot be certain about them.

As a result, many competent users instead revert to manual search, which can fully incorporate their prior knowledge. A recent survey showed that most NeurIPS 2019 and ICLR 2020 papers that reported having tuned hyperparameters used manual search, with only a very small fraction using BO (Bouthillier & Varoquaux, 2020). In order for BO to be adopted widely, and help facilitate faster progress in the ML community by tuning hyperparameters faster and better, it is therefore crucial to devise a method that fully incorporates expert knowledge into BO. In this paper, we introduce Prior-guided Bayesian Optimization (PrBO), a novel BO variant that combines user prior knowledge with a probabilistic model of the observations made. Our technical contributions with PrBO are:

1. For the first time, user prior knowledge can be combined with standard BO probabilistic models, such as Gaussian Processes (GPs), Random Forests (RFs), and Bayesian Neural Networks.
2. Unlike previous approaches, PrBO is flexible w.r.t. how the prior is defined, allowing previously impossible-to-inject (e.g. decay and exponential) priors.
3. PrBO gives more importance to the model as iterations progress, gradually forgetting the prior and ensuring robustness against misleading priors.

We demonstrate the effectiveness of PrBO on a comprehensive set of real-world applications and synthetic benchmarks, showing that accurate prior knowledge helps PrBO to achieve similar performance to current state-of-the-art on average $12.12\times$ faster on synthetic benchmarks and $1.49\times$ faster on a real-world application. PrBO also achieves equal or better final performance in all but one of the benchmarks tested.

2 BACKGROUND

2.1 BAYESIAN OPTIMIZATION

Bayesian Optimization (BO) is an approach for optimizing an unknown function $f : \mathcal{X} \to \mathbb{R}$ that is expensive to evaluate over an input space $\mathcal{X}$. In this paper, we aim to minimize f, i.e., find $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. BO approximates $\boldsymbol{x}^*$ with an optimal sequence of evaluations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \in \mathcal{X}$, with each new $\boldsymbol{x}_{n+1}$ depending on the previous function values $y_1, y_2, \ldots, y_n$ at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. BO achieves this by building a posterior on f based on the set of evaluated points. At each BO iteration, a new point is selected and evaluated based on the posterior, and the posterior is updated to include the new point $(\boldsymbol{x}_{n+1}, y_{n+1})$.

The points explored by BO are dictated by the acquisition function, which attributes a value to each $\boldsymbol{x} \in \mathcal{X}$ by balancing the predicted value and uncertainty of the prediction for each $\boldsymbol{x}$. In this work, as the acquisition function we choose Expected Improvement (EI) (Mockus et al., 1978), which quantifies the expected improvement over the best function value found so far: $EI_{f_{inc}}(\boldsymbol{x}) := \int_{-\inf}^{\inf} \max(f_{inc} - y, 0) p(y|\boldsymbol{x}) dy$, where f_{inc} is the incumbent function value, i.e., the best objective function value found so far, and $p(y|\boldsymbol{x})$ is a probabilistic model, e.g., a GP. Alternatives to EI would be Probability of Improvement (PI) (Jones, 2001), upper-confidence bounds (UCB) (Srinivas et al., 2010), and entropy-based methods (e.g. Hernández-Lobato et al. (2014)).

2.2 TREE-STRUCTURED PARZEN ESTIMATOR

Whereas the standard probabilistic model in BO directly models $p(y|\boldsymbol{x})$, the Tree-structured Parzen Estimator (TPE) approach of Bergstra et al. (2011) models $p(\boldsymbol{x}|y)$ and $p(y)$ instead. This is done by constructing two parametric densities, $g(\boldsymbol{x})$ and $l(\boldsymbol{x})$, which are computed using the observations with function value above and below a given threshold, respectively. The separating threshold y^* is defined as a quantile of the observed function values. TPE uses the densities $g(\boldsymbol{x})$ and $l(\boldsymbol{x})$ to define $p(\boldsymbol{x}|y)$ as:

$$p(\boldsymbol{x}|y) = l(\boldsymbol{x}) I(y < y^*) + g(\boldsymbol{x})(1 - I(y < y^*)), \tag{1}$$

where $I(y < y^*)$ is 1 when $y < y^*$ and 0 otherwise. The parametrization of the generative model $p(\boldsymbol{x}, y) = p(\boldsymbol{x}|y)p(y)$ facilitates the computation of EI as it leads to $EI_{y^*}(\boldsymbol{x}) \propto l(\boldsymbol{x})/g(\boldsymbol{x})$ and, thus, $\arg\max_{\boldsymbol{x}\in\mathcal{X}} EI_{y^*}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}\in\mathcal{X}} l(\boldsymbol{x})/g(\boldsymbol{x})$.

3 BAYESIAN OPTIMIZATION WITH PRIORS

We propose a BO approach dubbed PrBO that allows field experts to inject user prior knowledge into the optimization in the form of priors. PrBO combines this user-defined prior with a probabilistic model that captures the likelihood of the observed data $(\boldsymbol{x}_i, y_i)_{i=1}^n$. PrBO is independent from the probabilistic model being used, i.e., it can be freely combined with, e.g., GPs, RFs, or Bayesian Neural Networks.

3.1 PRIORS

PrBO allows users to inject prior knowledge into BO. This is done via a prior distribution that informs where in the input space $\mathcal{X}$ we expect to find good $f(\boldsymbol{x})$ values. A point is considered "good" if it leads to low function values. We denote the prior distribution $P_g(\boldsymbol{x})$, where g denotes that this is a prior on good points and $\boldsymbol{x} \in \mathcal{X}$ is a given point. Similarly, we define a prior on where in the input space we expect to have "bad" points. Although we could have a separate user-defined probability distribution $P_b(\boldsymbol{x})$, we aimed to keep the load on users low and thus, for simplicity, only require the definition of $P_g(\boldsymbol{x})$ and compute $P_b(\boldsymbol{x}) = 1 - P_g(\boldsymbol{x})$.[1] $P_g(\boldsymbol{x})$ is normalized to $[0, 1]$ by min-max scaling before computing $P_b(\boldsymbol{x})$.

In practice, $\boldsymbol{x}$ contains several dimensions but it is difficult for domain experts to provide a multivariate distribution $P_g(\boldsymbol{x})$. Users can easily specify, e.g., draw, a univariate or bivariate probability distribution for continuous dimensions or provide a list of probabilities for discrete dimensions. In PrBO, users are free to define a complex multivariate distribution, but we expect the standard use case to be that users only want to specify univariate distributions, implicitly assuming a prior that factors as $P_g(\boldsymbol{x}) = \prod_{i=1}^{D} P_g(x_i)$, where D is the number of dimensions in $\mathcal{X}$ and x_i is the i-th input dimension of $\mathcal{X}$. We show examples of continuous and discrete priors in Appendices B and F, respectively. We use factorized priors in our experiments to mimic what we expect most users will provide. In Appendix G, we show that these factorized priors lead to similar performance compared to multivariate priors.

3.2 MODEL

Whereas the standard probabilistic model in BO, e.g., a GP, quantifies $p(y|\boldsymbol{x})$ directly, that model is hard to combine with the user-defined prior $P_g(\boldsymbol{x})$. We therefore now introduce a method to translate the standard probabilistic model $p(y|\boldsymbol{x})$ into a model that is easier to combine with this prior. Similar to the TPE work described in Sec. 2.2, our generative model combines $p(\boldsymbol{x}|y)$ and $p(y)$ instead of directly modeling $p(y|\boldsymbol{x})$.

The computation we perform for this translation is to quantify the probability that a given input $\boldsymbol{x}$ is "good" under our standard probabilistic model $p(y|\boldsymbol{x})$. As in TPE, we define configurations as "good" if their observed y-value is below a certain quantile γ of the observed function values (so that $p(y < f_\gamma) = \gamma$). We in addition exploit the fact that our standard probabilistic model $p(y|\boldsymbol{x})$ has a Gaussian form, and under this Gaussian prediction we can compute the probability $\mathcal{M}_g(\boldsymbol{x})$ of the function value lying below a certain quantile using the standard closed-form formula for PI (Kushner, 1964):

$$\mathcal{M}_g(\boldsymbol{x}) = p(f(\boldsymbol{x}) < f_\gamma|\boldsymbol{x}) = \Phi\left(\frac{f_\gamma - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}}\right), \tag{2}$$

where $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ are the mean and standard deviation of the probabilistic model at $\boldsymbol{x}$, and Φ is the standard normal CDF, see Figure 1. Likewise, we compute a probability $\mathcal{M}_b(\boldsymbol{x})$ of $\boldsymbol{x}$ being bad.

3.3 PSEUDO-POSTERIOR

PrBO combines the prior in Section (3.1) and the model in Eq. (2) into a pseudo-posterior on "good" points. This pseudo-posterior represents the updated beliefs on where we can find good points, based on the prior and data that has been observed. The pseudo-posterior is computed as the product of the prior and the model:

$$g(\boldsymbol{x}) \propto P_g(\boldsymbol{x})\mathcal{M}_g(\boldsymbol{x})^{\frac{t}{\beta}}, \tag{3}$$

where t is the current optimization iteration, β is an optimization hyperparameter, $\mathcal{M}_g(\boldsymbol{x})$ is defined in Eq. (2), and $P_g(\boldsymbol{x})$ is the prior defined in Sec 3.1, rescaled to [0, 1]. We note that this pseudo-posterior is not normalized, but this suffices for PrBO to determine the next $\boldsymbol{x}_t$ as the normalization constant cancels out (c.f. Section 3.4). Since $g(\boldsymbol{x})$ is not normalized and we include the exponent t/β in Eq. 3, we refer to $g(\boldsymbol{x})$ as a pseudo-posterior, to emphasize that it is not a standard posterior probability distribution.

[1] We note that for continuous spaces, this $P_b(\boldsymbol{x})$ is not a probability distribution, and therefore only a pseudo-prior, as it does not integrate to 1. For discrete spaces, we normalize $P_b(\boldsymbol{x})$ so that it sums to 1 and therefore is a proper probability distribution and prior.

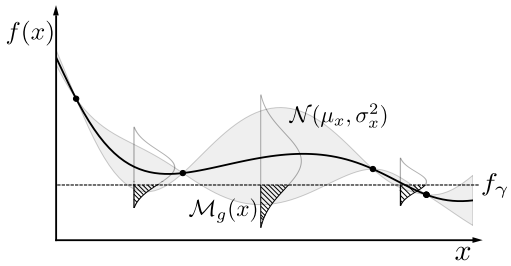


Figure 1: Our model is composed by a probabilistic model and the probability of improving over the threshold f_γ, i.e., right tail of the Gaussian. The black curve is the probabilistic model's mean and the shaded area is the model's variance.

Algorithm 1 PrBO Algorithm. $\boldsymbol{D}$ keeps track of all function evaluations so far: $(\boldsymbol{x}_i, y_i)_{i=1}^t$.

Input: Input space $\mathcal{X}$, user-defined prior distributions $P_g(\boldsymbol{x})$ and $P_b(\boldsymbol{x})$, quantile γ and BO budget B.
Output: Optimized point $\boldsymbol{x}_{inc}$.
$\boldsymbol{D} \leftarrow Initialize(\mathcal{X})$
for $t = 1$ **to** B **do**
  $\mathcal{M}_g(\boldsymbol{x}) \leftarrow fit_model_good(\boldsymbol{D})$
  $\mathcal{M}_b(\boldsymbol{x}) \leftarrow fit_model_bad(\boldsymbol{D})$
  $g(\boldsymbol{x}) \leftarrow P_g(\boldsymbol{x}) \cdot \mathcal{M}_g(\boldsymbol{x})^{\frac{t}{\beta}}$
  $b(\boldsymbol{x}) \leftarrow P_b(\boldsymbol{x}) \cdot \mathcal{M}_b(\boldsymbol{x})^{\frac{t}{\beta}}$
  $\boldsymbol{x}_t \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} EI_{f_\gamma}(\boldsymbol{x})$
  $y_t \leftarrow f(\boldsymbol{x}_t)$
  $\boldsymbol{D} = \boldsymbol{D} \cup (\boldsymbol{x}_t, y_t)$
end for
$\boldsymbol{x}_{inc} \leftarrow ComputeBest(\boldsymbol{D})$
return $\boldsymbol{x}_{inc}$

The t/β fraction in Eq. (3) controls how much weight is given to the model. As the optimization progresses, more weight is given to the model over the prior. Intuitively, we put more emphasis on the model as it observes more data and becomes more accurate. We do this under the assumption that the model will eventually be better than the user at predicting where to find good points. This also allows to recover from misleading priors as we show in Section 4.2; similar to, and inspired by Bayesian models, the data ultimately washes out the prior. The β hyperparameter defines the balance between prior and model, with higher β values giving more importance to the prior and requiring more data to overrule it.

We note that, computing Equation (3) directly can lead to numerical issues. Namely, the pseudo-posterior can reach extremely low values if the prior and model probabilities are low, especially as the t/β exponent grows. To prevent this, in practice, PrBO uses the logarithm of the pseudo-posterior instead: $\log(g(\boldsymbol{x})) \propto \log(P_g(\boldsymbol{x})) + \frac{t}{\beta} \cdot \log(\mathcal{M}_g(\boldsymbol{x}))$.

Once again, we also define an analogous pseudo-posterior distribution on bad $\boldsymbol{x}$: $b(\boldsymbol{x}) \propto P_b(\boldsymbol{x})\mathcal{M}_b(\boldsymbol{x})^{\frac{t}{\beta}}$. We then use these quantities to define a density model $p(\boldsymbol{x}|y)$ as follows:

$$p(\boldsymbol{x}|y) \propto \begin{cases} g(\boldsymbol{x}) & \text{if } \; y < f_\gamma \\ b(\boldsymbol{x}) & \text{if } \; y \geq f_\gamma. \end{cases} \tag{4}$$

3.4 Acquisition Function

We adopt the EI formulation used in (Bergstra et al., 2011) by replacing their Adaptive Parzen Estimators with our computation of the pseudo-posterior in Eq. (3). Namely, we compute EI as:

$$EI_{f_\gamma}(\boldsymbol{x}) := \int_{-\inf}^{\inf} \max(f_\gamma - y, 0) p(y|\boldsymbol{x}) dy = \int_{-\inf}^{f_\gamma} (f_\gamma - y) \frac{p(\boldsymbol{x}|y)p(y)}{p(\boldsymbol{x})} dy \propto \left(\gamma + \frac{b(\boldsymbol{x})}{g(\boldsymbol{x})}(1-\gamma)\right)^{-1}. \tag{5}$$

The full derivation of Eq. (5) is shown in Appendix C. Eq. (5) shows that to maximize improvement we would like points $\boldsymbol{x}$ with high probability under $g(\boldsymbol{x})$ and low probability under $b(\boldsymbol{x})$, i.e., minimizing the ratio $b(\boldsymbol{x})/g(\boldsymbol{x})$. We note that the point that minimizes the ratio for our unnormalized pseudo-posteriors will be the same that minimizes the ratio for the normalized pseudo-posterior and, thus, the computation of the normalized pseudo-posteriors is unnecessary.

The dynamics of the PrBO algorithm can be understood in terms of the following proposition (proof in Appendix C):

Proposition 1 *Given f_γ, $P_g(\boldsymbol{x})$, $P_b(\boldsymbol{x})$, $\mathcal{M}_g(\boldsymbol{x})$, $\mathcal{M}_b(\boldsymbol{x})$, $g(\boldsymbol{x})$, $b(\boldsymbol{x})$, $p(\boldsymbol{x}|y)$, and β as above, then*

$$\lim_{t\to\infty} \arg\max_{\boldsymbol{x}\in\mathcal{X}} EI_{f_\gamma}(\boldsymbol{x}) = \lim_{t\to\infty} \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathcal{M}_g(\boldsymbol{x}),$$

where EI_{f_γ} is the Expected Improvement acquisition function as defined in Eq. 5 and $\mathcal{M}_g(\boldsymbol{x})$ is as defined in Equation (2).

In early BO iterations the prior will have a predominant role, but in later BO iterations the model will grow more important, and as Proposition 1 shows, if PrBO is run long enough the prior washes out and PrBO *only* trusts the probabilistic model informed by the data.

3.5 Putting It All Together

Algorithm 1 shows the PrBO algorithm, based on the components defined in the previous sections. In Line 3, PrBO starts with a design of experiments (DoE) phase, where it randomly samples a number of points from the user-defined prior $P_g(\boldsymbol{x})$. After initialization, the BO loop starts at Line 4. In each loop iteration, PrBO fits the probabilistic model on the previously evaluated points (lines 5 and 6) and computes the pseudo-posteriors $g(\boldsymbol{x})$ and $b(\boldsymbol{x})$ (lines 7 and 8 respectively). The EI acquisition function is computed next, using the pseudo-posteriors, and the point that maximizes EI is selected as the next point to evaluate at line 9. The black-box function evaluation is performed at Line 10. This BO loop is repeated for a pre-defined number of iterations, according to the user-defined budget B. In Appendix A, we show a visualization of PrBO on a 1-dimensional function.

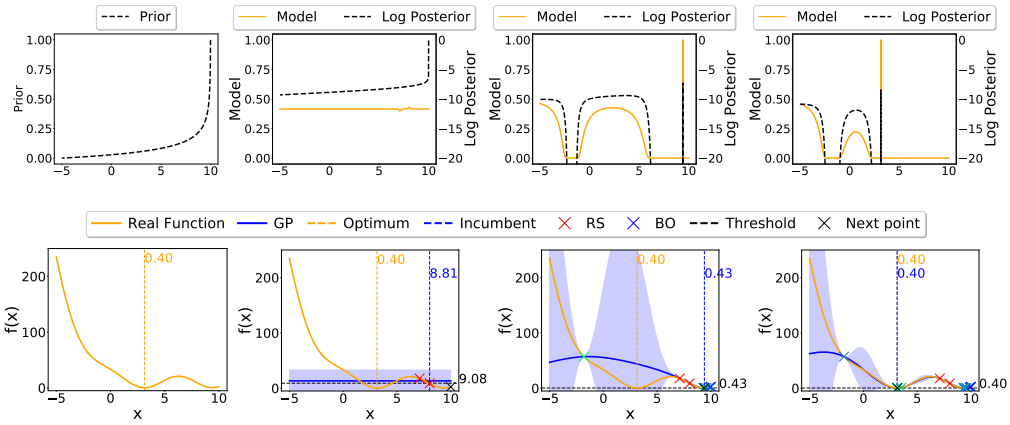
4 Experiments

We implement both GPs and RFs as predictive models and use GPs in our experiments unless stated otherwise. We set the model weight $\beta = 10$ and the model quantile to $\gamma = 0.05$, see our sensitivity hyperparameter study in Appendices L and M. Before starting the main BO loop in PrBO, we randomly sample $D + 1$ points from the prior consistently on all benchmarks. In Appendix J, we compare PrBO to other baselines with the same prior initialization and show that it leads to similar results. We optimize our EI acquisition function using a multi-start local search, similar to SMAC (Hutter et al., 2011). We start with four synthetic benchmarks: Branin, SVM, FC-Net, and XGBoost, which are 2, 2, 6, and 8 dimensional, respectively. The last three are part of the Profet benchmarks (Klein et al., 2019), generated by a generative model built using performance data on OpenML or UCI datasets. See Appendix D for more details on the experimental setup. Due to space constraints, we defer the (qualitatively similar) results for the SVM benchmark to Appendix E.

4.1 Prior Selection

In this section we study the effect of choosing a prior. A suitable property of the prior is that, by selecting a tighter prior around an optimum, we would expect sampling from the prior to have an increased performance. To the limit, if the prior is composed by only one point which is one of the global optima, then the first sample (and all of them) from the prior will hit the optimum. To have a sanity check of this property, we build an artificial prior in a controlled way. We rely on an automated computation of the prior by computing a univariate Kernel Density Estimation (KDE) using a Gaussian kernel on the synthetic benchmarks introduced above. We note that the goal of these synthetic priors is to have an unbiased prior for our experiments, whereas manual priors would be biased by our own expertise of these benchmarks. In practice, users will manually define these priors without needing additional experiments. Appendix G shows that using univariate KDEs for the prior lead to similar performance compared to multivariate KDEs.

We experiment with an array of varying quality priors. We select a constant $10D$ points in each prior and vary the size of the random sample dataset so that we can make the priors more sharply peaked around the optima in a controlled environment. We use the best performing $10D$ samples to create the prior from a uniform random sample dataset size of $10D\frac{100}{x}$; we refer to this prior as $x\%$ in Figure 3. As an example the XGBoost benchmark has $d = 8$, so, 100% means we sample 80 points and use all 80 to create the prior, 10% means we sample 800 points and use the best performing 80 to create the prior, 1% means we sample $8{,}000$ and use the best 80 to create the prior, and so on.



(a) No samples (b) 0 BO iterations (c) 10 BO iterations (d) 20 BO iterations

Figure 2: PrBO on the 1D Branin function. The leftmost column shows the exponential prior. The other columns show the model and the log pseudo-posterior after 0 (RS only), 10, and 20 BO iterations. PrBO forgets the wrong prior on the local optimum and converges to the global optimum.

Figure 3 shows the performance of purely sampling from the prior and running PrBO, respectively, after $10D$ function evaluations with different priors. A bigger random sample dataset and a smaller percentage leads to a tighter prior around the optimum, making the argument for a stronger prior. This is confirmed by Figure 3, where a sharply peaked prior (right side of the figure) leads to a better performance in both scenarios. In addition we observe that in contrast to sampling from the prior, PrBO achieves a smaller regret by being able to evolve from the initial prior and making independent steps towards better values of the objective function. More extensive experiments with a similar trend, including the rest of the benchmarks, are in Appendix K.

4.2 Prior Forgetting

In this section, we show that PrBO can recover from a misleading prior, thanks to our predictive model and the t/β parameter in the pseudo-posterior computation Eq. 3. As BO progresses, the predictive model becomes more accurate and receives more weight, guiding optimization away from the wrong prior and towards better values of the objective function. Figure 2 shows PrBO on the 1D Branin function with an exponential prior. Columns (b), (c), and (d) show PrBO after $D + 1 = 2$ initial samples and 0, 10, 20 BO iterations, respectively. After initialization, as shown in Column (b), the pseudo-posterior is nearly identical to the exponential prior and guides PrBO towards the region of the space on the right, which is towards the local optimum. This happens until the predictive model becomes certain there will be no more improvement from sampling that region (Columns (c) and (d)). After that, the predictive model guides the pseudo-posterior towards exploring regions with high uncertainty. Once the global minimum region is found, the pseudo-posterior starts balancing exploiting the global minimum and exploring regions with high uncertainty, as shown in 2d (bottom). Notably, the pseudo-posterior after $x > 4$ falls to 0 in 2d (top), as the predictive model is certain there will be no improvement from sampling the region of the local optimum. We provide additional examples of forgetting in Appendix B, and a comparison of PrBO with misleading priors, no prior, and correct priors in Appendix H.

4.3 Comparison Against Strong Baselines

Figure 4 compares PrBO to other optimizers using the log simple regret on five runs (mean and std error reported) on the synthetic benchmarks. We compare the results of PrBO with and without priors (both weak and strong) to 10,000× random search (RS, i.e., for each BO sample we draw 10,000 uniform random samples), sampling from the strong prior only, Spearmint (Snoek et al., 2012) and SMAC (Hutter et al., 2011) which are well-adopted BO approaches using the EI acquisition function and both GPs and RFs surrogate models, respectively.

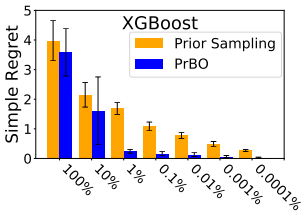


Figure 3: Regret of prior sampling and PrBO with different priors ($\mu \pm \sigma$ on 5 reps.).

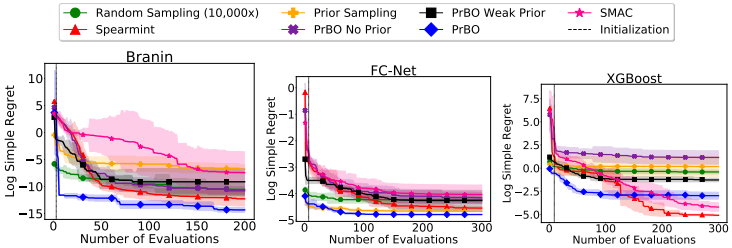


Figure 4: Log regret comparison of PrBO with and without priors, RS, and Spearmint (mean +/- one std on 5 repetitions). We run the benchmarks for $100D$ iterations, capped at 300.

PrBO prior beats $10{,}000\times$ RS and PrBO weak prior on all benchmarks. It also outperforms the performance of sampling from the prior; this is expected because prior sampling cannot recover from a non-ideal prior. The two methods are identical up to the initialization phase because they both sample from the same prior in that phase.

PrBO Prior is more sample efficient and finds better or equal results than Spearmint and SMAC on three out of the four benchmarks. On the fourth benchmark, XGBoost, PrBO leads the performance until 139 BO iterations, where Spearmint catches up and achieves slightly better results in the end (function values of 8.986 for Spearmint vs. 9.026 for PrBO after 300 iterations). Spearmint outperforms SMAC in all experiments. Importantly, in all our experiments, PrBO with a good prior consistently shows tremendous speedups in the early phases of the optimization process, typically only requiring on average 8.25 iterations to reach the performance that Spearmint reaches after 100 iterations ($12.12\times$ faster) and 5.5 iterations to reach the performance that SMAC reaches after 100 iterations ($18.18\times$ faster). Thus, in comparison to other traditional BO approaches, PrBO makes use of the best of both worlds, leveraging prior knowledge and efficient optimization based on BO.

4.4 The Spatial Use-case

We next apply PrBO to the `Spatial` (Koeplinger et al., 2018) real-world application. `Spatial` is a programming language and corresponding compiler for the design of application accelerators, i.e., FPGAs. We apply PrBO to three `Spatial` benchmarks, namely, 7D shallow and deep CNNs, and a 10D molecular dynamics grid application. We compare the performance of PrBO to RS, manual optimization, and HyperMapper (Nardi et al., 2019), the current state-of-the-art BO solution for `Spatial`. For a fair comparison, both PrBO and HyperMapper use RFs as predictive probabilistic model. The manual optimization and the prior for PrBO were provided by an unbiased `Spatial` developer, who is not an author of this paper. The priors were provided once and kept unchanged for the whole project. More details on the setup, including the priors used, are presented in Appendix F.

Figure 5 shows the log regret on the `Spatial` benchmarks. PrBO vastly outperforms RS in all benchmarks; notably, RS does not improve over the default configuration in MD Grid. PrBO is also able to leverage the expert's prior and outperform the expert's configuration in all benchmarks ($2.68\times$, $1.06\times$, and $10.4\times$ speedup for shallow CNN, deep CNN, and MD Grid, respectively). Compared to HyperMapper, PrBO achieves better performance in the MD Grid benchmark ($1.28\times$ speedup). For context, this is a significant improvement in the FPGA field, where a 10% improvement could qualify for acceptance in a top-tier conference. In the CNN benchmarks, PrBO converges to the minima regions faster than HyperMapper ($1.58\times$ and $1.4\times$ faster for shallow and deep respectively). Thus, PrBO leverages the best of both worlds (the expert's prior knowledge and BO) to provide a new state of the art for `Spatial`.

5 Related Work

TPE by Bergstra et al. (2011), the default optimizer in the popular HyperOpt package (Bergstra et al., 2013), supports limited hand-designed priors in the form of normal or log-normal distributions. We make three technical contributions that make PrBO more flexible than TPE. First, we generalize over

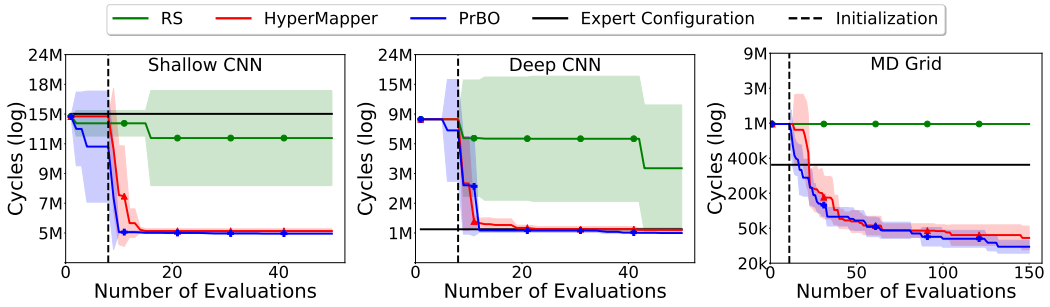


Figure 5: Log regret comparison of RS, HyperMapper, PrBO, and manual optimization on Spatial. The line and shaded regions show mean and std after 5 repetitions. Vertical lines are initialization.

the TPE approach by allowing more flexible priors; second, our approach is model-agnostic (i.e., PrBO is not limited to the TPE model; we use GPs and RFs in our experiments); and third, PrBO is inspired by Bayesian models that give more importance to the data as iterations progress. We also show that PrBO outperforms HyperOpt's TPE in Appendix I.

In parallel work, Li et al. (2020) allow users to specify priors via a probability distribution, similarly to our approach. Their two-level approach first samples a number of configurations by maximizing Thompson samples from a GP pseudo-posterior and then chooses the configuration that has the highest prior as the next to evaluate. In contrast, our method leverages the information from the prior more directly and ensures that the prior gets washed out as we collect more data, enabling PrBO to overcome misspecified priors.

Similarly, black-box optimization tools, such as SMAC (Hutter et al., 2011) or iRace (López-Ibáñez et al., 2016) also support simple hand-designed priors, e.g. log-transformations. However, these are not properly reflected in the predictive models and both cannot explicitly recover from bad priors. The same applies to the works of Oh et al. (2018) and Siivola et al. (2018).

Our work also relates to meta-learning for BO (Vanschoren, 2019), where BO is applied to many similar optimization problems in a sequence such that knowledge about the general problem structure can be exploited in future optimization problems. In contrast to meta-learning, PrBO is the first method that allows human experts to explicitly specify their priors. Furthermore, PrBO does not depend on any meta-features (Feurer et al., 2015b) and only incorporates a single prior instead of many priors from different experiments (Lindauer & Hutter, 2018).

6 Conclusions and Future Work

We have proposed a novel BO variant, PrBO, that allows users to inject their expert knowledge into the optimization in the form of priors about which parts of the input space will yield the best performance. These are different than standard priors over functions which are much less intuitive for users. BO failed so far to leverage the experience of domain experts, not only causing inefficiency but also getting users away from applying BO approaches because they could not exploit their years of knowledge in optimizing their black-box functions. PrBO addresses this issue and will therefore facilitate the adoption of BO. We showed that PrBO is 12.12x more sample efficient than state-of-the-art methods, and 10,000$\times$ faster than random search, on a common suite of benchmarks and achieves a new state-of-the-art performance on a real-world hardware design application. We also showed that PrBO converges faster and that it robustly recovers from misleading priors.

In future work, we will study how our approach can be used to leverage prior knowledge from meta-learning. Bringing these two worlds together will likely boost the performance of BO even further.

REFERENCES

Javad Azimi, Ali Jalali, and Xiaoli Zhang Fern. Hybrid batch bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML*. icml.cc / Omnipress, 2012.

Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable bayesian optimization in pytorch. *arXiv preprint arXiv:1910.06403*, 2019.

J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, 2013.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pp. 2546–2554, 2011.

Bruno Bodin, Luigi Nardi, M Zeeshan Zia, Harry Wagstaff, Govind Sreekar Shenoy, Murali Emani, John Mawer, Christos Kotselidis, Andy Nisbet, Mikel Lujan, et al. Integrating algorithmic parameters into benchmarking and design space exploration in 3d scene understanding. In *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, pp. 57–69, 2016.

Xavier Bouthillier and Gaël Varoquaux. Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020. Research report, Inria Saclay Ile de France, January 2020. URL `https://hal.archives-ouvertes.fr/hal-02447823`.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in alphago. *CoRR*, abs/1812.06855, 2018.

Laurence Charles Ward Dixon. The global optimization problem: an introduction. *Toward global optimization*, 2:1–15, 1978.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1436–1445, 2018.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2962–2970. Curran Associates, Inc., 2015a.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 1128–1135, 2015b.

Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on Machine Learning, ICML*, 2014.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 918–926. Curran Associates, Inc., 2014.

F. Hutter, L. Xu, H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer, 2011.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (eds.). *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR*, 2015.

Aaron Klein, Zhenwen Dai, Frank Hutter, Neil D. Lawrence, and Javier Gonzalez. Meta-surrogate benchmarking for hyperparameter optimization. In *Advances in Neural Information Processing Systems NeurIPS*, pp. 6267–6277, 2019.

David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, and Kunle Olukotun. Spatial: A Language and Compiler for Application Accelerators. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, June 2018.

Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Antonio Robles-Kelly, and Svetha Venkatesh. Incorporating expert prior knowledge into experimental design via posterior sampling. *arXiv preprint arXiv:2002.11256*, 2020.

Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 1355–1362, 2018.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 2242–2249. IEEE, 2012.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

Luigi Nardi, Bruno Bodin, Sajad Saeedi, Emanuele Vespa, Andrew J Davison, and Paul HJ Kelly. Algorithmic performance-accuracy trade-off in 3d vision applications using hypermapper. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1434–1443. IEEE, 2017.

Luigi Nardi, David Koeplinger, and Kunle Olukotun. Practical design space exploration. In *27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS*, 2019.

Arshak Navruzyan, Frank Sharp, Jeremy Howard, and Antoine Saliou. Optimizing hyperparams for image datasets in fastai. `https://platform.ai/blog/page/1/optimizing-hyperparams-for-image-datasets-in-fastai/`, 2019.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

C. Oh, E. Gavves, and M. Welling. BOCK : Bayesian optimization with cylindrical kernels. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3865–3874. PMLR, 2018.

Andrei Paleyes, Mark Pullin, Maren Mahsereci, Neil Lawrence, and Javier González. Emulation of physical processes with emukit. In *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Eero Siivola, Aki Vehtari, Jarno Vanhatalo, Javier González, and Michael Riis Andersen. Correcting boundary over-exploration deficiencies in bayesian optimization with virtual derivative sign observations. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2960–2968, 2012.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 1015–1022. Omnipress, 2010.

Joaquin Vanschoren. Meta-learning. In *Automated Machine Learning - Methods, Systems, Challenges*, pp. 35–61. 2019.